УДК 621.59 (075.8)

С.С. Сурков, А.Н. Мартынюк
Одесский национальный политехнический университет, просп. Шевченка, 1, Одесса, 65000, Украина

АВТОРИЗАЦИЯ АВТОМОБИЛЬНОГО КОМПЬЮТЕРА, НЕ ИМЕЮЩЕГО ПОДДЕРЖКИ БРАУЗЕРА, С МОБИЛЬНЫМИ УСТРОЙСТВАМИ ПОСРЕДСТВОМ ТЕХНОЛОГИИ BLUETOOTH

Вместе с взрывным ростом информационных технологий стремительно развивается коммуникабельность общества. Один из ярких примеров и доказательство тому – многочисленные социальные сети. Все автомобильные производители стремятся предоставить встраиваемым компьютерам доступ к Интернету, но некоторые встраиваемые компьютеры не имеет встроенного браузера. Чтобы решить проблему авторизации сторонних приложений был создан протокол OAuth, смысл которого заключается в вводе учетных данных в Web-браузере. Эта статья описывает решение реализации OAuth для встраиваемых компьютеров, которые не имеют поддержки браузера
Ключевые слова: *Метод авторизации – Автомобильный компьютер – OAuth – REST API – Смартфон – iOS – Android*

С.С. Сурков, А.Н. Мартинюк
Одеський національний політехнічний університет, просп. Шевченка, 1, Одеса, 65000, Україна

АВТОРИЗАЦІЯ АВТОМОБІЛЬНОГО КОМПЬЮТЕРА, ЯКИЙ НЕ МАЄ ПІДТРИМКИ БРАУЗЕРА, З МОБІЛЬНИХ ПРИСТРОЇВ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ BLUETOOTH

Разом з вибуховим зростанням інформаційних технологій стрімко розвивається комунікабельність суспільства. Один з яскравих прикладів і доказ тому – численні соціальні мережі. Всі автомобільні виробники прагнуть надати вбудованим комп'ютерам доступ до Інтернету, але деякі вбудовані комп'ютери не мають вбудованого браузера. Щоб вирішити проблему авторизації сторонніх додатків був створений протокол OAuth, сенс якого полягає в введенні облікових даних в Web-браузері. Ця стаття описує рішення реалізації OAuth для вбудованих комп'ютерів, які не мають підтримки браузера
Ключові слова: *Метод авторизації – Автомобільний комп'ютер – OAuth – REST API – Смартфон – iOS – Android*

DOI: 10.15673/0453-8307.2/2015.35890

I. ВВЕДЕНИЕ

Вместе с взрывным ростом информационных технологий стремительно развивается коммуникабельность общества. Один из ярких примеров и доказательство тому – многочисленные социальные сети, объединяющие большую часть жителей планеты, массово используемые автономные, интеллектуальные гаджеты. В тоже время мощный информационный поток, постоянно обрушивающий на членов общества, приводит к коллизиям и даже рискам для жизни. Так во многих странах правила дорожного движения запрещают во время движения пользоваться смартфонами, что требует появления специализированных клиентских решений.

Следует отметить, что создание интерактивных клиентских приложений под разные платформы и сетевые сервисы является важным. В частности, все более актуализируется возможность мобильного доступа водителя и пассажиров транспортных средств к социальным сетям и сервисам.

Как известно, для доступа такого рода приложения к REST API[5, 8] веб-сервиса необходимо обеспечивать безопасность работы, в частности, осуществлять авторизацию. Существует множество реализаций авторизации, которые также позволяют зашифровать данные пользователя, передаваемые через сеть, простейший из которых – передача вместо данных хэшей данных.

Для более полного решения проблемы безопасности сторонних приложений был предложен протокол OAuth [4, 7] (рисунок 1) основное назначение которого заключается в том, чтобы выполнять ввод данных пользователя через стандартный веб-браузер, в результате получить токен доступа, который используется для последующего доступа к REST API.

Описание протокола приведено в работе [1], исследование его на предмет атак выполнено в работе [2], где также показана интеграция OAuth в сервис TeraGrid и отмечены достоинства и недостатки протокола, в работе [3] специально исследованы атаки на веб-сайты, использующие технологию OAuth и формальные методы.

Усилением протокола oAuth является протокол авторизации xAuth, в котором включены преимущества авторизации oAuth и шифрования данных, а также гарантируется, что данные не будут переданы злоумышленнику во время обмена с сервером. Однако, здесь существует опасность, что стороннее приложение украдет данные, так как при предоставлении сервисом REST API возможности разработки сторонних клиентов не устраняется проблема того, что в самом клиенте могут быть реализованы вредоносные функции, такие как отправка данных пользователя злоумышленнику.

В общем можно отметить, что OAuth - это современный, достаточно надежный протокол для авторизации в веб сервисах сторонних пользовательских приложений, являющийся стандартом де факто. Его используют такие корпорации как Google, Facebook, Twitter, FourSquare, ВКонтакте и многие другие, в связи с чем его можно выбрать в качестве базового протокола для последующего расширения и усовершенствования. Примером проблемы, предполагающей такое развитие, может быть отсутствие веб-браузера в существующих встраиваемых автомобильных компьютерах.

Почти все веб-сервисы имеют собственные приложения для передовых мобильных платформ, например, таких как Android и iOS, которые авторизуются по протоколу XAuth с использованием данных пользователя.

Ведущие производители автомобилей стараются создать удобства для своих пользователей разработкой встраиваемых автомобильных компьютеров, поэтому все их современные модели имеют доступ в интернет, который используется для REST[6] базируемых приложений, однако не все модели авто компьютеров имеют веб-браузер. Таким образом возникает проблема создания REST приложения, которое будет авторизоваться посредствам OAuth.

Стандартная последовательность действий авторизации с помощью протокола OAuth может быть представлена в виде UML-диаграммы последовательностей (рисунок 1).

Здесь пункты 3-6 диаграммы предусматривают необходимость наличия веб браузера для использования протокола OAuth.

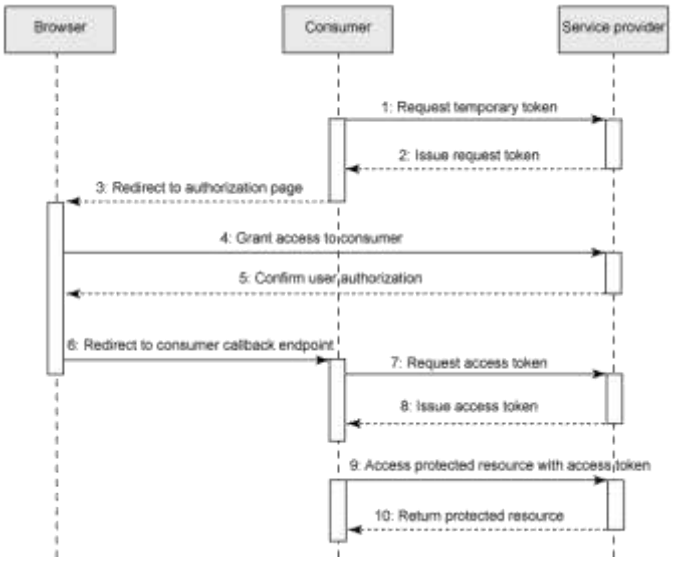

Рисунок 1 – Диаграмма работы протокола OAuth

Исследование известных решений позволяет выявить их достоинства и недостатки. Сразу можно отметить, что для веб-сервисов предоставление REST-API является стандартом де факто для легкого создания родных приложений любой платформы, имеющей доступ в интернет.

Вместе с тем, у сторонних разработчиков нет доступа к серверной стороне, который мог бы обеспечить обход метода авторизации. Особую сложность доставляет тот факт, что много автомобилей уже эксплуатируются, и установка браузера будет весьма трудоемким процессом как для разработчиков, так и для водителей.

Для решения задачи авторизации без веб браузера можно пойти несколькими путями. Известен метод реверс инжиниринга и передача данных на сервер что передает веб браузер. Данный подход будет хорошо работать, но есть риск что через некоторое время веб-сервис изменит алгоритм авторизации и веб-приложение не сможет работать.

Другой метод предполагает интеграцию и адаптацию веб браузера во встраиваемый компьютер. Этот метод является самым очевидным, но

для его реализации необходимо адаптировать веб-браузер под встраиваемый компьютер и требуется большее количество затрат на его поддержку. Но самое главное препятствие – это сложность установки браузера во встраиваемые компьютеры, которые находятся на дороге.

Также известен метод разработки прокси сервиса, который использует авторизацию с использованием логина и пароля и предоставляет доступ к API сервиса через свои промежуточные API. В этом случае проблема состоит в том, что необходимо после окончания проекта длительно поддерживать данный сервис, результатом чего будет лишнее расходование людских ресурсов.

Известны реализации прокси сервисов, которые предоставляют доступ, но у них есть недостаток в том, что они используют другие API, отличные от веб сервиса, также есть небольшой риск в том, что на стороне прокси сервиса могут возникнуть проблемы, в этом случае исчезают достоинства OAuth, показаны выше.

Никакой из указанных методов авторизации OAuth качественно не справляется с задачей, в них существует либо риск того, что провайдер изменит схему авторизации, что в современных условиях довольно частое явление, либо существует задача длительной поддержки продукта или проблемы адаптации и интеграции веб браузера.

В этой связи возникает необходимость создания нового подхода к авторизации, который бы не имел недостатков отказа либо длительной поддержки продукта.

II. ПОСТАНОВКА ЗАДАЧИ

Целью настоящего исследования является разработка метода построения коммуникационных мобильных систем на основе протокола OAuth для встраиваемых автомобильных компьютеров, не обладающих собственными веб-браузерами.

Для достижения этой цели необходимо решить задачу выбора оптимальных стратегий авторизации для встраиваемого компьютера без поддержки браузера.

Для решения данной задачи необходимо решить несколько подзадач:

- создание структурной схемы метод авторизации
- построение модели взаимодействий в виде диаграмм:
  a) протокола авторизации
  b) состояний авторизации OAuth через Bluetooth
  c) работы протокола авторизации OAuth через Bluetooth.
- практическая реализация метода на примере Android приложения и приложения для встраиваемого компьютера

III. ПОСТРОЕНИЕ МЕТОДА

Решение данной задачи дает предлагаемый в данной работе метод, основанный на использовании браузера смартфона для выполнения всей браузерной части (рисунок 2).

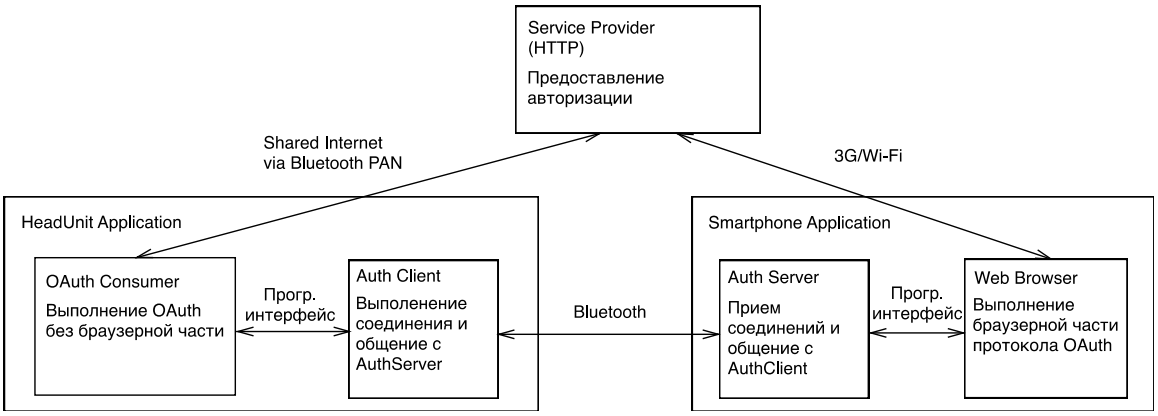

Рисунок 2 – Структурная схема авторизации протокола OAuth через Bluetooth

Компонент «Service Provider» является поставщиком OAuth авторизации он же обычно предоставляет REST-API, компонент «OAuth Consumer» выполняет авторизацию без браузерной части и веб-браузер выполняет браузерную часть авторизации. Для этого в него необходимо передать URL полученный от OAuth потребителя.

В данном решении, по сравнению с оригинальной схемой добавляются два компонента: Auth Client и Auth Server, задача которых заключается в выполнении соединения и коммуникацией между друг другом.

Как видно для пунктов 3-6 из рисунка 1 для передать URL в браузерную часть и получить ответ потребителя назад. При ошибке соединения происходит сброс в начальное состояние как смартфонного как компьютерного приложения.

Из последовательности действий протокола OAuth (рис. 1) видно, что веб-браузеру передается перенаправление на авторизацию, после чего браузерная часть продолжает работу с поставщиком сервиса и передает ответ потребителю. Это делает логичным возможность оставить общение потребителя и поставщика сервиса на встраиваемом

компьютере, а часть выполняемую веб браузером выполнить на смартфоне. Токен передается встраиваемому компьютеру посредствам Bluetooth.

Данный решение может работать как на iOS (необходимо сертифицированное устройство MFi), так и на Android, в будущем планируется расширить список платформ. Для реализации метода на основе предлагаемого решения разработан протокол авторизации для устройств без поддержки браузера (рисунок 3).

Метод предполагает следующие действия:

Для начала работы протокола необходимо установления Bluetooth соединения по протоколу SPP (Serial Port Profile) с приложением которое находится на смартфоне.

Далее приложение на встраиваемом компьютере передает команду для получения Access Token'а, принимая ее приложение на смартфоне загружает страницу авторизации.

После успешной авторизации веб браузер передает Access Token приложению смартфона, которое в свою очередь передает приложению на встраиваемом компьютере.

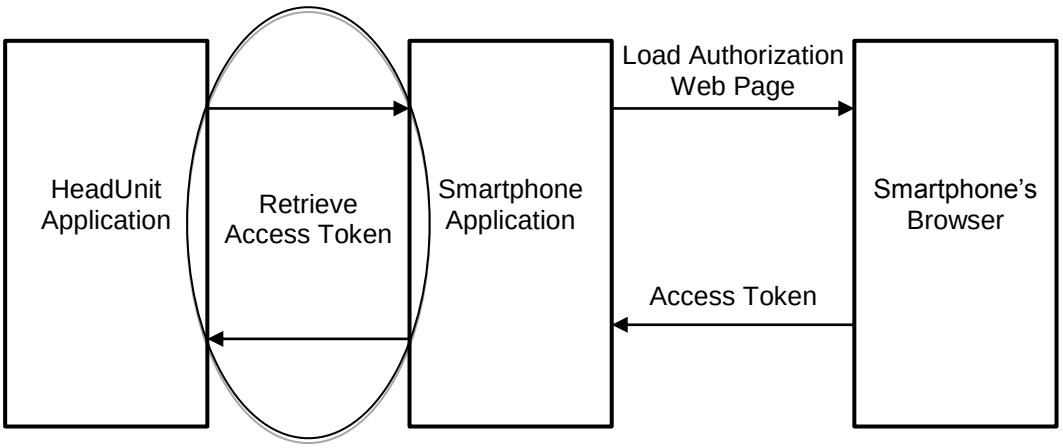

Рисунок 3 – Диаграмма протокола авторизации для устройств без поддержки браузера.

Как видно, основу метода составляет алгоритм авторизации встраиваемого компьютера через браузер смартфона (рисунок 4). В процедуре для соединения встраиваемого компьютера и смартфона встраиваемый компьютер проверяет, есть ли спаренные телефоны по технологии Bluetooth, после чего устанавливается подключение к приложению смартфона по протоколу SPP.

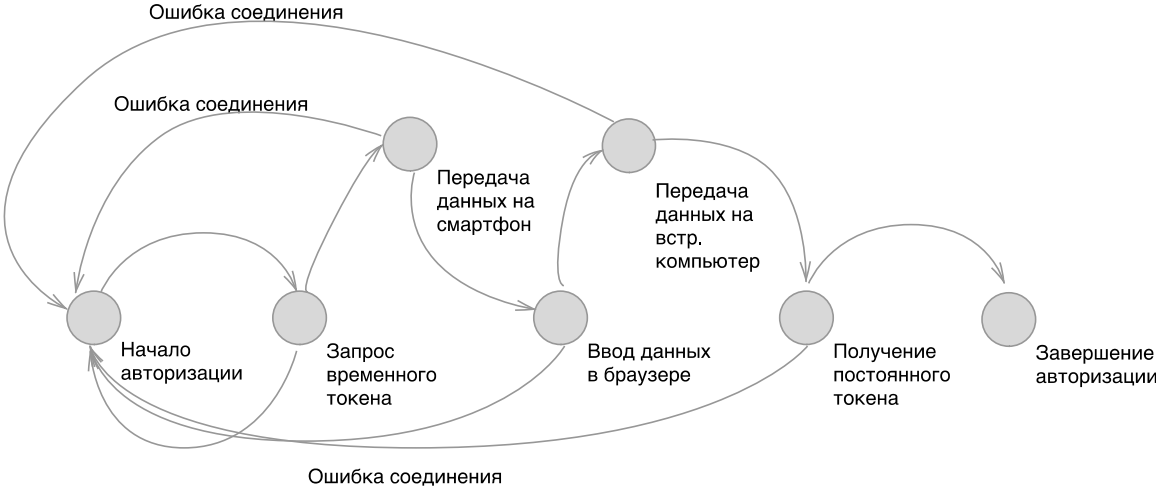

Рисунок 4 – Диаграмма состояний работы протокола авторизации OAuth через Bluetooth.

В работе протокола авторизации выделяются три основных состояния (рисунок 4):

- запрос временного токена;
- запрос данных в браузере;
- получение постоянного токена.

Из вершины запроса временного токена происходит передача адресной строки со встраиваемого компьютера на смартфон.

После вершины «ввод данных в браузере» выполняется обратная передача через вершины «Передача данных на смартфон» и «Передача данных на встраиваемый компьютер».

Если происходит ошибка соединения как с поставщиком сервиса, так и Bluetooth соединения, то происходит возврат в вершину начала авторизации.

При успешном завершении авторизации приложение может отправлять запросы к REST API.

При успешном подключении на экране смартфона появляется OAuth диалог, в котором пользователь вводит свои логин и пароль, и в случае успешной авторизации встраиваемому компьютеру возвращается токен социальной сети.

SPP соединение остается открытым, когда оба устройства включены. Соединение завершается при условиях успешного получения токена, когда одно из приложений прекращает работу либо происходит таймаут соединения.

Таким образом, в результате настоящей работы предложена модификация метода авторизации протокола OAuth, в которой выполняется авторизация на стороне смартфона. Диаграмму последовательностей модифицированного метода можно увидеть на рисунке 5.

В результирующей диаграмме добавились два компонента: Auth Server и Auth Client через которые встроенный компьютер связывается со смартфоном.

В начале процесса добавляются шаги передача URL переадресации компоненту AuthClient и установление соединения с AuthServer (3-4). На данных шагах может произойти исключительная ситуация разрыва соединения Bluetooth и для обработки ее необходимо продумать дополнительные меры.

После авторизации в браузере выполняются шаги (8-9), в которых AuthServer передает данные встраиваемому компьютеру.

Во время авторизации пользователя риск разрыва Bluetooth соединения увеличивается и необходимо реализовать логику обработки ошибок и восстановления соединения. Для обработки ошибок, которые могут возникать на стороне браузера, следует обрабатывать по возможность на стороне смартфона, и в случае критической ошибки отправлять сообщение на смартфон.

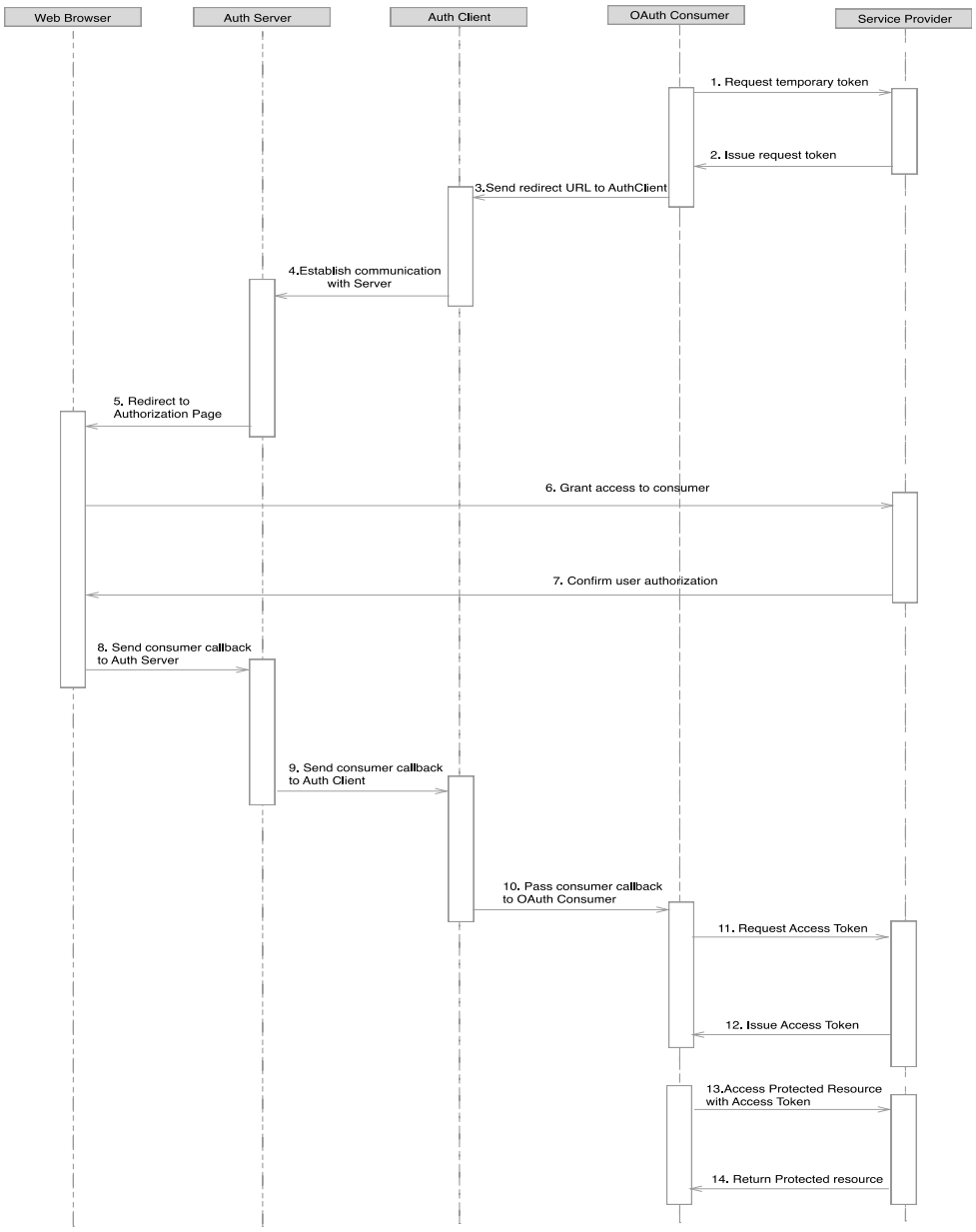

Рисунок 5 – Диаграмма работы протокола авторизации OAuth через Bluetooth.

IV. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Для проверки работоспособности метода было разработано приложение на платформе Android которое позволяет авторизовать приложения для встраиваемого компьютера: Facebook, Google+, Evernote. UML диаграмма данного приложения показана на рисунке 6:

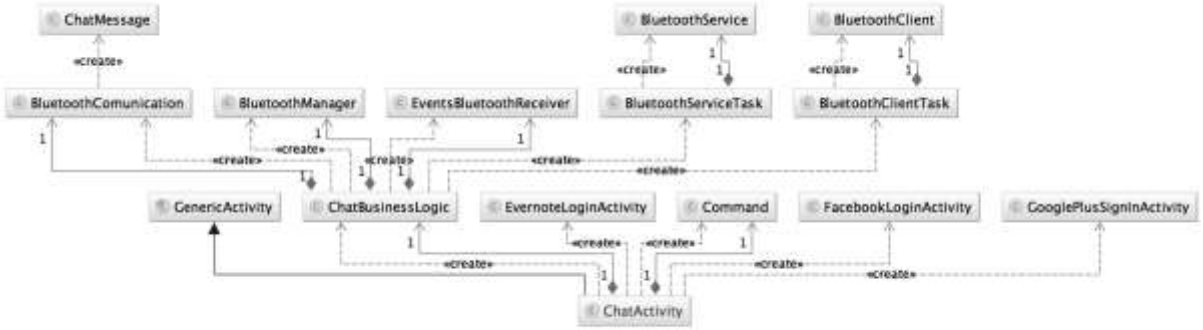

Рисунок 6 – UML диаграмма Android приложения.

При старте приложения запускается ChatActivity которая создает при событии пользователя ожидает входящее подключение по каналу Bluetooth. Для обработки входящих подключений запускается Bluetooth Service Task, которая предназначена для выполнения UI событий, в то время как Bluetooth Service устанавливает соединение по каналу SPP.

Также рассматривается вариант с использованием смартфона как клиента. Для этого используются классы BluetoothClient и BluetoothClientTask и EventBluetoothReceiver для получения списка устройств. В результате успешного выполнения асинхронной операции выполняется объект BluetoothSocket, из которого можно получить InputStream и OutputStream, которые использует класс BluetoothCommunication отвечающий за прием и передачу сообщений через сокет.

Для передачи сообщений используется формат JSON[9] и для запуска activity для необходимой авторизации используется необходимо приложению на смартфоне послать команду вида:

```
{"app":"appName",
"command":"startOAuthDialog"}
```

После обработки команды запускается необходимое Activity и запускается процесс авторизации и после успешной авторизации передается JSON команда на встраиваемый компьютер, после чего смартфонное приложение можно закрыть.

Многие социальные сети предоставляют свои SDK для интеграции своих сервисов в мобильные приложения. Это значительно упрощает добавление функциональности в приложение для смартфона. Так же интеграция SDK в проект позволяет авторизоваться через мобильные клиенты без необходимости ввода данных пользователя. Данная возможность позволяет существенно повысить удобство авторизации пользователя очень сильно уменьшив время на его вход и оставаясь при этом надежным и безопасным способом авторизации.

Метод авторизации через смартфон был проверен экспериментально. На рисунке 4 показана схема обмена данными между смартфонным и приложением для встраиваемого компьютера. В начале встраиваемый компьютер устанавливает соединение по протоколу SPP и смартфонное приложение отравляет команду что оно готово к работе. Далее приложение для встраиваемого компьютера посылает запрос на получение токена и переходит в состояние ожидания токена.

После получения токена посылается команда на сервер что работа завершена и смартфонное приложение больше не нужно.

Данный метод был внедрен при разработке приложений под iOS на Objective C и для Android на Java.

Полезный эффект состоит в том что нет необходимости разрабатывать собственный веб браузер, можно использовать веб браузер смартфона для создания REST приложений для которых требуется OAuth авторизация.

Данный метод позволяет разработать серию программ, представляющих новые возможности для пользователей, которые не могли быть реализованы старыми методами.

V. ЗАКЛЮЧЕНИЕ

Проведенное исследование показало, что комбинация данных технологий открывает новые возможности и позволяет разрабатывать новые приложения в основе которых лежит данный метод. Эти приложения позволяют получить доступ к REST API социальных сервисов которые используют OAuth.

Такую комбинацию можно рекомендовать для устройств, которые не поддерживают веб-браузер и у которых имеется поддержка технологии Bluetooth.

ЛИТЕРАТУРА

1. **E. Hammer-Lahav (ed.).** The OAuth 1.0 Protocol. IETF RFC 5849 (Informational), April 2010. http://tools.ietf.org/html/rfc5849.
2. **Jim Basney and Jeff Gaynor** An OAuth Service for Issuing Certificates to Science Gateways for TeraGrid Users, National Center for Supercomputing Applications University of Illinois at Urbana-Champaign 1205 West Clark Street, Urbana, Illinois 61801.
3. **Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Sergio Maffeis.** Discovering Concrete Attacks on Website Authorization by Formal Analysis, April 2013.
4. OAuth 2.0: The Definitive Guide O'Reilly Media; 1 edition (25 Jun 2014) ISBN-10: 1449319319.
5. **Jim Webber, Savas Parastatidis, Ian Robinson** REST in Practice Hypermedia and Systems Architecture By O'Reilly Media September 2012.
6. **Leonard Richardson, Sam Ruby** RESTful Web Services Web services for the real world O'Reilly Media May 2011.
7. **LeBlanc J.** - Programming Social Applications: Building Viral Experiences with OpenSocial, OAuth, OpenID 2011
8. **Masse Mark.** REST API Design Rulebook O'Reilly Media, 2013.
9. **T. Bray, Ed.**, The JavaScript Object Notation (JSON) Data Interchange Format, IETF RFC 7159, March 2014, http://tools.ietf.org/html/rfc7159.
10. **R. Fielding, J. Reschke**, Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, IETF RFC 7230, June 2014, https://tools.ietf.org/ html/ rfc7230.

S.S. Surkov, A.N. Martynyuk
Odessa National Polytechnic University, av. Shevchenko, 1, Odessa, 65000, Ukraine

AUTHORIZATION FOR AUTOMOBILE HEADUNIT WITHOUT BROWSER SUPPORT WITH MOBILE DEVICES THROUGH BLUETOOTH

Along with the crucial growth of information-technologies the communicability of the society is rapidly developing. Multiple social networks is the brightest example of this fact. All automobiles manufacturers strive to provide embedded computers with access to the Internet, but some embedded computers does not have a built-in browser. To solve the problem of applications authorization protocol OAuth was created, the aim of which is to enter Web-browser accounting data. This paper describes the implementation of the OAuth realization solution for embedded computers that do not have browser support.
All social networks such as Facebook, Twitter, Google+, Foursquare, Evernote, VK provide WEB REST API which allows to create native application easily for any platform which have internet access.
The communication is going through Bluetooth. As access to internet is implemented via Bluetooth protocol there won't be technical difficulties to use the app. This method could be implemented as for iOS (headunit is needed to be MFI certified) or for Android.
This paper describes the solution of implementing OAuth for headunits which without browser support.
Keywords: *Authorization Method – Headunit – OAuth – REST API – Smartphone – iOS – Android*

REFERENCES

1. **E. Hammer-Lahav (ed.).** The OAuth 1.0 Protocol. IETF RFC 5849 (Informational), April 2010. http://tools.ietf.org/html/rfc5849.
2. **Jim Basney, Jeff Gaynor.** An OAuth Service for Issuing Certificates to Science Gateways for TeraGrid Users, National Center for Supercomputing Applications University of Illinois at Urbana-Champaign 1205 West Clark Street, Urbana, Illinois 61801.
3. **Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Sergio Maffeis.** 2013. Discovering Concrete Attacks on Website Authorization by Formal Analysis, April 2013.
4. OAuth 2.0: The Definitive Guide O'Reilly Media; 1 edition (25 Jun 2014) ISBN-10: 1449319319.
5. **Jim Webber, Savas Parastatidis, Ian Robinson.** 2012. REST in Practice Hypermedia and Systems Architecture By O'Reilly Media September 2012.
6. **Leonard Richardson, Sam Ruby.** 2011. RESTful Web Services Web services for the real world O'Reilly Media May 2011.
7. **LeBlanc J.** - Programming Social Applications: Building Viral Experiences with OpenSocial, OAuth, OpenID 2011.
8. **Masse Mark.** 2013. REST API Design Rulebook O'Reilly Media, 2013.
9. **T. Bray, Ed.** 2014. The JavaScript Object Notation (JSON) Data Interchange Format, IETF RFC 7159, March 2014, http://tools.ietf.org/html/rfc7159.
10. **R. Fielding, J. Reschke.** 2014. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, IETF RFC 7230, June 2014.

Отримана в редакції 15.01.2015, прийнята до друку 03.03.2015